What is claimed is:

1. A method of processing a series of data packets for transmission over a data network in a series of frames in which at least some of the frames contain multiple data packets, each data packet in the series of data packets having a respective time in a time sequence, each frame being capable of transmitting a certain amount of data, the method comprising:

successively joining data packets from the time sequence into the frames and transmitting each data packet in at least one of the frames no later than a certain time interval after the respective time of said each data packet in the time sequence, which includes

(a) transmitting each frame in a first set of the frames upon filling said each frame in the first set of frames with data from one or more of the data packets so that said each frame in the first set of frames cannot contain an additional data packet; and

(b) transmitting each frame in a second set of the frames which are not filled with at least some of the data packets so that said each frame in the second set of the frames cannot contain an additional data packet in order to ensure that said each data packet is transmitted in at least one of the frames no later than the certain time interval after the respective time of said each data packet in the time sequence.

2. The method as claimed in claim 1, wherein a main routine for processing said each data packet initiates the transmitting of each frame in the first set of the frames upon filling said each frame in the first set of frames with data from one or more of the data

packets so that said each frame in the first set of frames cannot contain an additional data packet; and

wherein a timer interrupt routine initiates the transmitting of each frame in the second set of the frames which are not filled with at least some of the data packets so that said each frame in the second set of the frames cannot contain an additional data packet in order to ensure that said each data packet is transmitted in at least one of the frames no later than the certain time interval after the respective time of said each data packet in the time sequence .

3. The method as claimed in claim 1, wherein the data packets include read I/O request data packets and write I/O request data packets, and the method includes separately joining the read I/O request data packets together for transmission, and separately joining the write I/O request data packets together for transmission, so that the I/O request data packets have an ordering in the frames that is different from the ordering of the I/O request data packets in the time sequence.

4. The method as claimed in claim 3, wherein some of the read I/O request data packets are moved in front of some of the write I/O request data packets in some of the frames.

5. The method of claim 1, wherein the data packets are I/O request data packets, and the method includes on-line transaction processing applications in a host processor producing the data packets, and a TCP/IP interface in the host processor transmitting the

frames over an IP network to network attached storage containing a database accessed by the on-line transaction processing applications.

6. The method of claim 1, wherein the data packets are I/O replies from network attached storage, and the frames are transmitted to a host processor accessing the network attached storage.

7. The method of claim 1, wherein the data packets are stored in a range of addresses of memory, a certain number of frames are preallocated in another region of memory, and the data packets are joined by transfer of the data packets from the range of addresses in memory to the preallocated frames in memory.

8. The method of claim 7, wherein the certain number of preallocated frames are periodically updated.

9. The method of claim 7, which includes application threads loading the data packets into the memory at the range of addresses in memory.

10. The method of claim 7, which includes TCP/IP threads accessing the pool of preallocated frames for transmission of the preallocated frames including the data packets over an IP network.

11. The method as claimed in claim 1, which includes transmitting the frames over a data network, measuring loading on the data network, and dynamically adjusting the duration of the certain time interval based on the measured loading of the data network, the duration of the certain time interval being increased for increased loading on the data network.

12. In a host processor programmed for executing on-line transaction processing applications and having a network block storage interface for accessing network attached storage coupled to the host processor via a data network, a method comprising the host processor joining I/O request data packets from different ones of the on-line transaction processing applications in the same network transmission frames to more completely fill the network transmission frames.

13. The method as claimed in claim 12, which includes the host processor transmitting each I/O request data packet in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

14. The method as claimed in claim 13, which includes the host processor dynamically adjusting the certain time interval in response to loading on the data network, the certain time interval being increased for increased loading on the data network.

15. The method as claimed in claim 12, which includes the host processor executing a periodic timer interrupt routine to insure that each I/O request data packet is transmitted in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

16. The method as claimed in claim 12, wherein the I/O request data packets include read I/O request data packets and write I/O request data packets, and the method includes separately joining the read I/O request data packets together for transmission to the network block storage, and separately joining the write I/O request data packets together for transmission to the network block storage.

17. The method as claimed in claim 16, which includes moving some of the read I/O request data packets in front of some of the write I/O request data packets in some of the frames.

18. The method as claimed in claim 12, which includes turning on and off the joining of the I/O request data packets.

19. The method as claimed in claim 12, wherein the joining of the I/O request data packets is turned off during a bulk transfer of database data.

20. The method as claimed in claim 12, which includes the host processor executing an I/O request bunching routine that intercepts I/O request data packets sent from the on-line transaction processing applications to a network block storage interface.

21. The method of claim 12, which includes storing the I/O request data packets in a range of addresses of memory, preallocating a certain number of frames in another region of memory, and joining the data packets during transfer of the data packets from the range of addresses in memory to the preallocated frames in memory.

22. The method of claim 21, which includes periodically updating the certain number of preallocated frames.

23. The method as claimed in claim 12, which includes the network attached storage bunching I/O replies into frames for transmission from the network attached storage over the data network to the host processor.

24. A method of solving a performance problem in a host processor programmed for executing on-line transaction processing applications and having a network block storage interface for accessing network attached storage coupled to the host processor via a data network, the performance problem being caused by network transmission frames being only partially filled with I/O request data packets from the on-line transaction processing applications, the performance problem being solved by re-programming the host processor to join the I/O request data packets from different ones of the on-line

transaction processing applications in the same network transmission frames to more completely fill the network transmission frames.

25. The method as claimed in claim 24, which includes re-programming the host processor to transmit each I/O request data packet in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

26. The method as claimed in claim 25, which includes re-programming the host processor for dynamic adjustment of the certain time interval in response to loading on the data network, the certain time interval being increased for increased loading on the data network.

27. The method as claimed in claim 24, wherein the re-programming of the host processor includes adding a periodic timer interrupt routine to insure that each I/O request data packet is transmitted in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

28. The method as claimed in claim 24, wherein the I/O request data packets include read I/O request data packets and write I/O request data packets, and the host processor is re-programmed for separately joining the read I/O request data packets together for

transmission to the network block storage, and separately joining the write I/O request data packets together for transmission to the network block storage.

29. The method as claimed in claim 28, wherein the host processor is reprogrammed to move some of the read I/O request data packets in front of some of the write I/O request data packets in some of the frames.

30. The method as claimed in claim 24, which includes re-programming the host processor for turning on and off the joining of the I/O request data packets.

31. The method as claimed in claim 24, wherein the host processor is re-programmed by adding an I/O request bunching module that intercepts I/O request data packets sent from the on-line transaction processing applications to a network block storage interface.

32. The method as claimed in claim 24, wherein the host processor is re-programmed by modifying programming in the network block storage interface that packs the frames with the I/O request data packets.

33. The method as claimed in claim 24, which includes re-programming the network attached storage to bunch I/O replies into frames for transmission from the network attached storage over the data network to the host processor.

34. A host processor programmed for executing on-line transaction processing applications and having a network block storage interface for accessing network attached storage coupled to the host processor via a data network, the host processor being programmed for joining the I/O request data packets from different ones of the on-line transaction processing applications into the same network transmission frames to more completely fill the network transmission frames.

35. The host processor as claimed in claim 34, wherein the host processor is programmed for transmitting each I/O request data packet in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

36. The host processor as claimed in claim 35, wherein the host processor is programmed for dynamically adjusting the certain time interval in response to loading on the data network, the certain time interval being increased for increased loading on the data network.

37. The host processor as claimed in claim 34, wherein the host processor is programmed with a periodic timer interrupt routine to insure that each I/O request data packet is transmitted in a frame no later than a certain time interval after said each I/O request data packet is produced by one of the on-line transaction processing applications.

38. The host processor as claimed in claim 34, wherein the I/O request data packets include read I/O request data packets and write I/O request data packets, and the host processor is programmed for separately joining the read I/O request data packets together for transmission to the network block storage, and for separately joining the write I/O request data packets together for transmission to the network block storage.

39. The host processor as claimed in claim 38, which is programmed for moving some of the read I/O request data packets in front of some of the write I/O request data packets in some of the frames.

40. The host processor as claimed in claim 34, wherein the host processor is programmed for turning on and off the joining of the I/O request data packets.

41. The host processor as claimed in claim 34, wherein the host processor is programmed with an I/O request bunching routine that intercepts I/O request data packets sent from the on-line transaction processing applications to the network block storage interface.

42. The host processor as claimed in claim 34, wherein the host processor is programmed for storing the I/O request data packets in a range of addresses of memory, preallocating a certain number of frames in another region of memory, and joining the data packets during transfer of the data packets from the range of addresses in memory to the preallocated frames in memory.

43. The host processor as claimed in claim 42, which is programmed for periodically updating the certain number of preallocated frames.